INVITED PAPER *Special Section on the 2002 IEICE Excellent Paper Award*

Requirement Specification and Derivation of ECA Rules for Integrating Multiple Dissemination-Based Information Sources**

Tomoyuki KAJINO[†*], ***Nonmember***, **Hiroyuki KITAGAWA**[††], ***and*** **Yoshiharu ISHIKAWA**[††], ***Members***

SUMMARY The recent development of network technology has enabled us to access various information sources easily, and their integration has been studied intensively by the data engineering research community. Although technological advancement has made it possible to integrate existing heterogeneous information sources, we still have to deal with information sources of a new kind—*dissemination-based information sources*. They actively and autonomously deliver information from server sites to users. Integration of dissemination-based information sources is one of the popular research topics. We have been developing an information integration system in which we employ ECA rules to enable users to define new information delivery services integrating multiple existing dissemination-based information sources. However, it is not easy for users to directly specify ECA rules and to verify them. In this paper, we propose a scheme to specify new dissemination-based information delivery services using the framework of relational algebra. We discuss some important properties of the specification, and show how we can derive ECA rules to implement the services.

key words: *information dissemination, information integration, ECA rule, relational algebra*

1. Introduction

The recent development of network technology has enabled us to access various information sources easily. Due to the demand for an integration facility of heterogeneous information sources, information integration (mediation) has become an important data engineering research issue [7], [12]. Although technological advancement has made it possible to integrate existing heterogeneous information sources, we must still deal with information sources of a new kind—*dissemination-based information sources*. They actively and autonomously deliver information from server sites to clients. Therefore, users can receive up-to-date information automatically, and do not have to worry about where the information is located or when it is updated. It is, however, not easy to integrate data from multiple dissemination-based information sources and conventional information sources, or to obtain the integration results when the users want them to be delivered.

We have been developing an information integration system which integrates various information sources including dissemination-based information sources [10]. The system provides two features regarding information dissemination: (1) Integration of dissemination-based information sources and (2) Dissemination-based delivery of the integration result. The former enables user-specified selection of the delivered information and integration with other information sources such as relational databases and Web pages. The latter enables effective delivery of the integrated data from the integration system based on the timing requirement specified by users. For example, it is possible to extract interesting portions from the disseminated information, integrate them with data in relational databases, and then periodically deliver the integration results to users using the dissemination facility.

These features require "active" facilities such as event handling and support for timing constraints. To realize this, we have employed ECA rules [13]. Users can define new information delivery services, if they add appropriate ECA rules to the system.

However, it is not an easy task for users to specify ECA rules. Moreover, ECA rules are related to each other, and it is difficult to verify their consistency. An approach to this problem is to provide a framework in which users can specify their integration requirements in a more declarative manner, in order to validate the consistency of the specifications, and to automatically generate ECA rules from the validated specifications.

In this paper, we propose a specification scheme to realize the framework. In the scheme, information sources including dissemination-based sources are modeled as relations, and new information delivery services are specified using relational algebra-based expressions. Then, the consistency of the specifications is examined and ECA rules for implementing the user requirements are derived from algebra-based expressions, and the in-

Manuscript received August 7, 2003.

[†]The author is with Master's Program in Sciences and Engineering, University of Tsukuba, Tsukuba-shi, 305–8573 Japan.

[††]The authors are with Institute of Information Sciences and Electronics, University of Tsukuba, Tsukuba-shi, 305–8573 Japan.

[*]Presently, Beacon Information Technology Inc.

[**]This article was originally published in the IEICE Transactions on Information and Systems (Japanese Edition), vol.J85-D-I, no.1, pp.40–52, January 2002.

tegration system provides the *information delivery service* by executing the ECA rules.

The remainder of this paper is organized as follows. In Sect. 2, we give a simple example of a new information delivery service integrating multiple existing dissemination-based information sources. The mediator/wrapper-based integration system architecture assumed in this paper is described in Sect. 3. Section 4 describes the ECA rules employed in our approach. In Sect. 5, we show how dissemination-based information sources are modeled as relations, and present a framework to specify the new information delivery services based on the relational algebra. In Sect. 6, we discuss some properties for verifying the given specifications. In Sect. 7, we show how we can derive ECA rules to implement the services from them. In Sect. 8, prototype system implementation is discussed. In Sect. 9, related works are mentioned. In Sect. 10, the conclusion and future works are presented.

2. Integration Example

In this section, we describe a simple integration scenario. We define a new information delivery service integrating dissemination-based information sources and a relational database (Fig. 1). We assume that there exist two dissemination-based information sources. *Industrial news service (INews)* sends industrial news articles to the clients using e-mails. Each e-mail message contains data items such as the company name, news header, and contents. *Stock price information feed service (SPrice)* provides the closing stock price information everyday. Namely, it sends a record that consists of the company name, its category of business, and the closing stock price of the day.

In addition to these dissemination-based information sources, we assume a relational database, namely, *Stockholder's information database (StockholderInfo)*. This database contains a relation that manages information on the stock items that a stockholder owns. The relation has attributes such as the company name, the purchase price, the number of stocks, and a threshold price value specified by the owner.

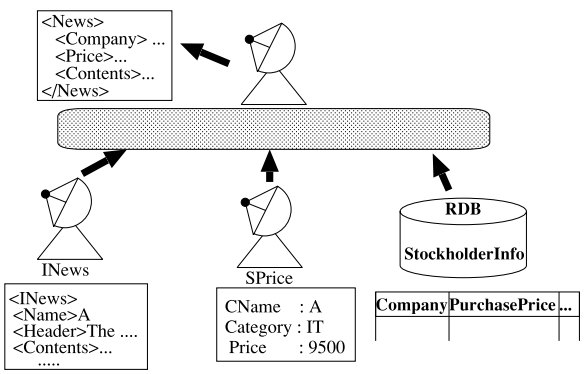

Fig. 1 Integration example.

Here, we assume a user owns stocks of companies in the IT category, and the stock information is stored in the Stockholder's information database. To show an information integration example, assume that the user has the following demands:

- When a new closing stock price for an IT category company has arrived, check it and send a notification message to the user if it exceeds its *threshold value*. The threshold value is specified by the user for each IT stock item and stored as an attribute value in the Stockholder's information database.
- The notification message should include the closing stock price, company name, and news contents related with the company extracted from the industrial news articles delivered in the last two days.
- The notification should be sent to the user at midnight on that day.

The above demands can be met by defining a new information delivery service in which notification messages generated from data in the underlying three information sources are periodically sent to the user. We show how this requirement is specified in our scheme in Sect. 5.

3. Integration Architecture

In our research, we assume a mediator/wrapper-based information integration system architecture [7], [10], [12]. The relational model is used here as a common data model at the mediator level, since the model is well established and provides a sound technical basis. To deal with incoming data from dissemination-based information sources, we allocate wrappers, named *DIS wrappers*, to these sources in addition to wrappers for relational databases and Web pages. This system integrates data from the underlying information sources, triggered by events such as the arrival of new data and time progress. The integration results can be actively delivered from the system to users through new dissemination-based information services.

The system employs ECA rules to specify such event-driven data integration and delivery actions. The system architecture is shown in Fig. 2. A DIS wrapper receives information (*delivery units*) sent from a dissemination-based information source, and raises events to notify the arrival of delivery units. The *rule processing module* holds the ECA rules, which specify actions to be triggered by events raised by DIS wrappers and the *timer module*. When events are raised, the rule processing module executes relevant ECA rules, which invoke data storage, integration, and delivery actions. The *mediator* is responsible for data integration. It also manages temporary relations for storing delivery units. The *dissemination module* is responsible for the delivery of the integration results obtained from the

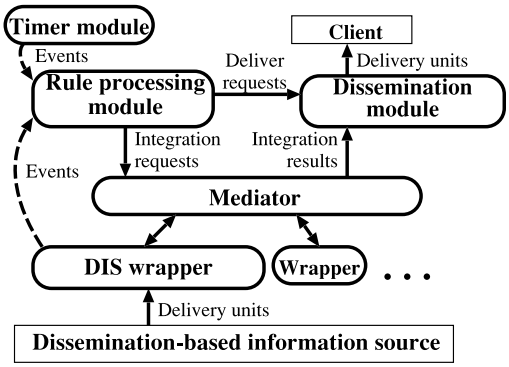

Fig. 2 Integration system architecture.

mediator.

Using the integration example in Sect. 2, we illustrate the function of each module.

1. When the DIS wrapper receives a delivery unit from the industrial news service, the wrapper translates it into a tuple in the relational model, and notifies an arrival event to the rule processing module.
2. The rule processing module invokes an ECA rule corresponding to the notified event, and sends a request to the mediator to store the new data in a temporary relation.
3. When a delivery unit from the stock price information feed service has arrived, the DIS wrapper also raises an arrival event.
4. The rule processing module invokes another ECA rule, and requests the mediator to check whether the delivered data is related to an IT category company. If it is, the mediator stores the new data in another temporary relation, and sets the timer alarm for midnight of that day.
5. At midnight, the timer module raises an event. Then, the rule processing module requests the mediator to integrate the stock price information and the industrial news articles related with the company which were delivered in the last two days, if the stock price exceeds the specified threshold value.
6. Finally, the rule processing module orders the dissemination module to deliver the integration result to the user.

4. ECA Rules

In this section, we explain ECA rules and classify them into three categories from the viewpoint of their roles in our context. Basically, ECA rules in this paper are the same as those in active databases. An ECA rule consists of three parts: the *event* (**on**), *condition* (**if**), and *action* (**do**) clauses.

The event clause specifies an event that triggers activation of the rule. There are two kinds of events: *primitive event* and *composite event*. As indicated in Sect. 3, the following two primitive events are raised from DIS wrappers and the timer module:

1. **arrival**(R): This event is raised from a DIS wrapper and notifies the arrival of a new delivery unit from the dissemination-based information source R.
2. **alarm**(*Alarm Name*): This event is raised from the timer module when the set time has come. *Alarm Name* designates each alarm event, and is given when the timer module alarm is set by the *setTimer* operator.

A composite event is constructed from primitive events and/or composite events.

The condition clause specifies a precondition which must be met for the action clause to be processed. In this context, we allow the condition clauses to contain selection conditions in relational algebra expressions.

The action clause specifies data storage, integration, and delivery actions. Basically, those actions are expressed in relational algebra expressions[†]. As mentioned in Sect. 3, the mediator manages temporary relations. Updates of temporary relations are denoted as follows.

(Temp) **:=** *(Expression)*:
Tuples *(Expression)* are assigned to a temporary relation *(Temp)*.
(Temp) **+=** *(Expression)*:
Tuples *(Expression)* are appended to a temporary relation *(Temp)*.
(Temp) **−=** *(Expression)*:
Tuples *(Expression)* are deleted from a temporary relation *(Temp)*.

In addition, the Deliver operator is used to send data delivery requests to the delivery module. More details of the Delivery operator are given in [10].

As explained in Sect. 3, various operations, such as data storage, data integration, and data delivery, are involved in the process of providing a new data delivery service based on dissemination-based information sources. We divide this process into the following three phases, and each phase is implemented as a set of ECA rules.

1. *Storage of delivery units*: In this phase, triggered by arrival events, the mediator checks whether data sent from the dissemination-based information sources are necessary to meet the user requirements. If they are, they are stored in temporary relations. In the example scenario in Sect. 3, steps 2 and 4 correspond to this phase. ECA rules employed to implement this phase are called *storage*

[†]Deliver and setTimer operators are also used as explained below.

rules.

2. *Generation of new delivery units*: In this phase, triggered by alarm events, the mediator extracts relevant data from temporary relations and generates the requested delivery units, integrating them with data in other information sources. Then, the delivery module is invoked for data delivery. In the example in Sect. 3, steps 5 and 6 correspond to this phase. ECA rules used for this phase are called *generation rules*.
3. *Disposal of unnecessary delivery units*: In this phase, unnecessary data in temporary relations are discarded. In the example in Sect. 3, stock price data stored in the temporary relation becomes obsolete after midnight. Also, industrial news articles become of no use after midnight, the following day. ECA rules used for this purpose are called *garbage disposal rules*, and their activations are triggered by alarm events.

Storage rules corresponding to steps 2 and 4, respectively, in the example in Sect. 3 can be written as follows.

Rule Storage$_{INews}$
on: arrival(I_{INews})
do: $Temp_{I_{INews}}$ += I_{INews};

Rule Storage$_{SPrice}$
on: arrival(I_{SPrice})
if: $I_{SPrice}.Category =' IT'$
do: $Temp_{I_{SPrice}}$ += I_{SPrice};
setTimer($next_{*:*:*:0:0:0}(I_{SPrice}.ITS)$, *new*);

Here, I_{INews} and I_{SPrice} denote the most recently delivered data unit (tuple) from the INews and SPrice information sources, respectively†. The *setTimer* operator in the action clause of the second rule sets the timer alarm for the time specified by $next_{*:*:*:0:0:0}(I_{SPrice}.ITS)$. The alarm event invokes a generation rule corresponding to steps 5 and 6. $I_{SPrice}.ITS$ denotes the arrival time of the SPrice data, and the function $next_{*:*:*:0:0:0}(I_{SPrice}.ITS)$ derives the time corresponding to midnight of that day. Details of such functions to derive designated time are explained in Sect. 5.

5. Declarative Requirement Specification

As mentioned in Sect. 3, we use the relational model as a common model at the mediator level. In this section, we show how to specify new information delivery services integrating dissemination-based information sources based on the relational algebra. In Sect. 7, we show how ECA rules can be derived from the specifications.

First, we model incoming data from dissemination-based information sources and outgoing data through new information delivery services as relations. Relations modeling incoming data from dissemination-based information sources are called *I-sequence relations*. Relations modeling outgoing data through new information delivery services are called *O-sequence relations*. In both sequence relations, tuples correspond to delivery units. Each I-sequence relation has a timestamp attribute ITS, which designates the arrival time of the corresponding delivery unit. Each O-sequence relation has a timestamp attribute OTS, which designates the (scheduled) delivery time of the corresponding delivery unit. In addition to ITS and OTS, they have their own attributes depending on the information sources and the delivery requests.

Logically, an I-sequence relation represents all the data delivered from the underlying dissemination-based information source from its service start time to the service end time. Of course, when the information source is still in service, the whole instance of the I-sequence relation cannot be materialized. Similarly, an O-sequence relation represents all the data delivered through the new information delivery service. By modeling incoming and outgoing data as I-sequence and O-sequence relations, respectively, we can define a new information delivery service by specifying how to obtain an O-sequence relation from I-sequence relations and, if necessary, conventional relations representing other information sources. The basic idea here is to employ the relational algebra for this purpose. The relational algebra provides a sound basis.

In our scheme, a new information delivery service is defined by the following formula:

$$O_{new} = \Omega_{f(I_k.ITS)}(E(I_1, \cdots, I_n)),$$

where O_{new} is an O-sequence relation and E is a relational algebra expression to derive a new relation from I-sequence relations $I_1, \cdots, I_n$ and other conventional relations. The operator $\Omega_{f(I_k.ITS)}$ is introduced here. It adds the OTS attribute, sets $f(I_k.ITS)$ as its value, and removes all the ITS attributes included in the relation obtained by E. The function f is a *timestamp function*, and derives a new timestamp from the ITS attribute value in the I-sequence relation I_k. I_k is called a *master I-sequence relation* (or *information source*), and must be referenced in E. Note that the OTS values are decided based on the ITS values in the master I-sequence relation.

The new information delivery service explained in Sect. 2 can be specified as follows:

$$\begin{aligned} O_{new} = \Omega_{next_{*:*:*:0:0:0}(I_{SPrice}.ITS)}(& \\ & \sigma_{Category=' IT'}(I_{SPrice}) \\ & \bowtie_{CName=Name \wedge previous_{*:*:*:0:0:0}(\\ & \quad before_{0:0:1:0:0:0}(I_{SPrice}.ITS) \end{aligned}$$

†The interpretation of symbols I_{INews} and I_{SPrice} in storage rules is slightly different from that in expressions in the other parts. Details are explained in Sect. 7.1.2.

$$\begin{aligned}
&\leq previous_{*:*:*:0:0:0}(I_{INews}.ITS))\\
&\wedge previous_{*:*:*:0:0:0}(I_{INews}.ITS)\\
&\quad \leq previous_{*:*:*:0:0:0}(I_{SPrice}.ITS)\\
&I_{INews}\\
&\bowtie_{Name=Company \wedge Price \geq Threshold}\\
&StockholderInfo).
\end{aligned}$$

Here, I_{INews} and I_{SPrice} are I-sequence relations, and $I_{INews}.ITS$ and $I_{SPrice}.ITS$ are their ITS attributes, respectively. The functions *next*, *previous*, and *before* are timestamp functions. Selection, projection, and join operations are represented by σ, π, and $\bowtie$, respectively.

Timestamp functions considered in the paper are as follows:

1. **immediate**(t): Returns the given timestamp t itself.
2. **next**$_p(t)$: Returns the timestamp, matching the temporal pattern p, chronologically next to the timestamp t.
3. **previous**$_p(t)$: Returns the timestamp, matching temporal pattern p, chronologically previous to the timestamp t.
4. **after**$_{\delta t}(t)$: Returns the timestamp after the time interval δt from t.
5. **before**$_{\delta t}(t)$: Returns the timestamp before the time interval δt from t.

The temporal pattern p is given in one of the following formats:

(1) *year:month:day:hour:minute:second*
(2) *year:month:dayofweek:hour:minute:second*

Nonnegative integers and the wild card symbol '*' can be specified for fields *year*, *month*, *day*, *hour*, *minute* and *second*†. The field *dayofweek* can contain one of the strings {Sun, Mon, Tue, Wed, Thu, Fri, Sat} or '*'. The time interval δt is specified using the format (1), but '*' is not allowed.

For example,

$$next_{*:*:*:0:0:0}(I_{SPrice}.ITS)$$

gives the timestamp corresponding to the next midnight following the arrival time $I_{SPrice}.ITS$.

$$\begin{aligned}
&previous_{*:*:*:0:0:0}(I_{INews}.ITS)\\
&= previous_{*:*:*:0:0:0}(I_{SPrice}.ITS)
\end{aligned}$$

checks whether the timestamps $I_{INews}.ITS$ and $I_{SPrice}.ITS$ imply the same day.

To simplify the discussion, we make the following assumptions in the remaining part of this paper.

1. No I-sequence relation is referenced in E more than once.
2. Only timestamp functions and primitive comparison operators ($=$, $\neq$, $<$, $>$, $\leq$, $\geq$) can be used in selection and join conditions concerning timestamps in E.

When an I-sequence relation is referenced more than once in E, we can deal with the case by regarding each instance as an independent I-sequence relation. Thus, the first assumption can be eliminated. (In this case, we need to perform some modification to the rule derivation explained in Sect. 7.) Although it is also possible to remove the second assumption, we keep it to make the discussion more concrete.

6. Basic Properties of Requirement Specifications

Unfortunately, users could specify infeasible information delivery services as follows:

$$\begin{aligned}
O_{new} = \Omega_{immediate(I_{SPrice}.ITS)}(&I_{SPrice}\\
&\bowtie_{CName=Name \wedge}\\
&previous_{*:*:*:0:0:0}(after_{0:0:1:0:0:0}(I_{SPrice}.ITS))\\
&=previous_{*:*:*:0:0:0}(I_{INews}.ITS)I_{INews}).
\end{aligned}$$

The expression specifies the following requirement:

> *As soon as the system receives a closing stock price data of some company, say A, from the stock price information feed source, it should deliver it with the next day's news articles related to A.*

Obviously, this is infeasible, since the expected news articles have not yet arrived when new delivery units are to be delivered. To exclude such infeasible specifications, we define the notion of consistency.

Before explaining the notion of consistency, we introduce the *inverse* $f^{-1}(t)$ for the timestamp function $f(t)$ as follows:

$$f^{-1}(t) = \{u | f(u) = t\}.$$

For each timestamp function introduced in Sect. 5, the inverse is defined as follows:

1. **immediate**$^{-1}$
   $immediate^{-1}(t) = \{u | u = t\}$
2. **next**$^{-1}$
   $next_p^{-1}(t) = \{u | previous_p(t) \leq u < t\}$
3. **previous**$^{-1}$
   $previous_p^{-1}(t) = \{u | t < u \leq next_p(t)\}$
4. **after**$^{-1}$
   $after_{\delta t}^{-1}(t) = \{u | u = before_{\delta t}(t)\}$
5. **before**$^{-1}$
   $before_{\delta t}^{-1}(t) = \{u | u = after_{\delta t}(t)\}$.

When the expression $f(t) = f_1(f_2(\cdots(f_n(t))\cdots))$ is given, $f^{-1}(t)$ is defined as follows:

$$\begin{aligned}
f^{-1}(t) = \{u | &(\exists u_{n-1}) \cdots (\exists u_1)(u \in f_n^{-1}(u_{n-1})\\
&\wedge \cdots \wedge u_2 \in f_2^{-1}(u_1) \wedge u_1 \in f_1^{-1}(t))\}.
\end{aligned}$$

†To simplify the discussion, we assume that '*' is always specified for *year*.

Definition 1: The specification $O_{new} = \Omega_{f(I_k.ITS)}(E(I_1, \cdots, I_n))$ is said to be consistent, if, for all t,

$$\sigma_{OTS=t}(\Omega_{f(I_k.ITS)}(E(I_1, \cdots, I_n))) = \sigma_{OTS=t}(\Omega_{f(I_k.ITS)}(E(\sigma_{ITS\leq t}(I_1), \cdots, \sigma_{ITS\leq t}(I_n)))).$$

□

This definition means that tuples to be delivered at time t must be generated only from tuples obtained up to the time t.

In our scheme, we check the consistency of a given specification based on the following theorem.

Theorem 1: If the specification $O_{new} = \Omega_{f(I_k.ITS)}(E(I_1, \cdots, I_n))$ satisfies the following conditions, it is consistent:

(i) The expression $\sigma_{I_k.ITS\in f^{-1}(t)}(E(I_1, \cdots, I_n))$ can be rewritten into the following form:

$$E'(\sigma_{I_1.ITS\in\psi_1(t)}(I_1), \cdots, \sigma_{I_n.ITS\in\psi_n(t)}(I_n)),$$

where $\psi_1(t), \cdots, \psi_n(t)$ give selection conditions regarding the ITS values of I_1, $\cdots$, I_n, respectively.

(ii) For each $i\,(1 \leq i \leq n)$, $\psi_i(t)$ has the upper bound $max(\psi_i(t))$, and it is always less than or equal to t: $max(\psi_i(t)) \leq t$, for any t value that OTS can take. □

Proof *In Definition 1, OTS corresponds to the formula $f(I_k.ITS)$ in the specification. Therefore, the left-hand side of the equation in Definition 1 can be transformed as follows:*

$$\begin{aligned}&\sigma_{OTS=t}(\Omega_{f(I_k.ITS)}(E(I_1, \cdots, I_n)))\\ &= \sigma_{f(I_k.ITS)=t}(\Omega_{f(I_k.ITS)}(E(I_1, \cdots, I_n)))\\ &= \Omega_{f(I_k.ITS)}(\sigma_{I_k.ITS\in f^{-1}(t)}(E(I_1, \cdots, I_n))). \quad (1)\end{aligned}$$

By Theorems 1 (i) and (ii), expression (1) can be transformed into

$$\Omega_{f(I_k.ITS)}(\sigma_{I_k.ITS\in f^{-1}(t)}(E(\sigma_{I_1.ITS\leq t}(I_1), \cdots, \sigma_{I_n.ITS\leq t}(I_n)))).$$

Replacing the selection condition $I_k.ITS \in f^{-1}(t)$ with $f(I_k.ITS) = t$, we obtain

$$\sigma_{OTS=t}(\Omega_{f(I_k.ITS)}(E(\sigma_{I_1.ITS\leq t}(I_1), \cdots, \sigma_{I_n.ITS\leq t}(I_n)))).$$

□

The rewriting in Theorem 1 (i) is performed by pushing down the selection conditions on ITS values as in the conventional query optimization process [16]. Theorem 1 (ii) assures that tuples to be delivered at time t can be generated only from tuples obtained up to the time t.

As mentioned in Sect. 4, we employ storage rules, generation rules, and garbage disposal rules. However, in some cases, we do not need garbage disposal rules. Let us consider the following requirement:

> *The system should deliver all the important news articles related to company A obtained in the past on its anniversary every year.*

The specification expressing this requirement is obviously consistent. If only this delivery service is defined and important news articles on company A are stored in a temporary relation, we cannot discard any information. In this case, we need no garbage disposal rule for the temporary relation. Definition 2 defines the property related to this. In Definition 2 and the remaining part of this paper, we use the following notations:

$$TSet(\psi_i, t) = \bigcup_{\tau>0} \psi_i(t+\tau),$$

$$TSet^{+}(\psi_i, t) = \bigcup_{\tau\geq 0} \psi_i(t+\tau).$$

Definition 2: Assume a consistent specification $O_{new} = \Omega_{f(I_k.ITS)}(E(I_1, \cdots, I_n))$ is given so that it can be rewritten into the form of Theorem 1 (i). Then, if

$$\psi_i(t) - TSet(\psi_i, t) \neq \phi$$

for some t, it is said that the I-sequence relation I_i may contain disposable data under ψ_i. □

As shown in the next section, garbage disposal rules are generated only for I-sequence relations which may contain disposable data.

7. Derivation of ECA Rules

In this section, we discuss derivation of ECA rules from the relational algebra-based specifications explained in Sect. 5. First, we show how to derive storage rules, generation rules, and garbage disposal rules from the specification of a new information delivery service in Sect. 7.1. Sect. 7.2 discusses cases where multiple information delivery services are specified.

7.1 Basic Derivation Scheme

7.1.1 Overview

For a specification $O_{new} = \Omega_{f(I_k.ITS)}(E(I_1, \cdots, I_n))$, the derivation of ECA rules is outlined as follows:

1. Transform the given specification into $\sigma_{I_k.ITS\in f^{-1}(t)}(E(I_1, \cdots, I_n))$.
2. Push down selection conditions as much as possible as in the conventional query optimization scheme, and rewrite the above expression in the following form:

   $$E'(\sigma_{C_1}(\sigma_{I_1.ITS\in\psi_1(t)}(I_1)), \cdots, \sigma_{C_n}(\sigma_{I_n.ITS\in\psi_n(t)}(I_n))),$$

   where $C_i\,(1 \leq i \leq n)$ is the selection condition for I_i's attributes except ITS.

3. Check whether the specification is consistent. If it is not, exit from the procedure.
4. Derive a storage rule for each I-sequence relation $I_i\,(1 \leq i \leq n)$.
5. Derive a generation rule.
6. For each I-sequence relation I_i, check whether it may contain disposable data under ψ_i. If it can, derive a garbage disposal rule for I_i.

When we apply steps 1 and 2 to the example specification given in Sect. 5, we obtain the following expression.

$$\begin{aligned}
&\sigma_{Category='IT'} \\
&\quad (\sigma_{I_{SPrice}.ITS \in \{u | previous_{*:*:*:0:0:0}(t) \leq u < t\}}(I_{SPrice})) \\
&\bowtie_{CName=Name \wedge previous_{*:*:*:0:0:0}(before_{0:0:1:0:0:0})} \\
&\quad (I_{SPrice}.ITS) \leq previous_{*:*:*:0:0:0}(I_{INews}.ITS) \\
&\quad \wedge previous_{*:*:*:0:0:0}(I_{INews}.ITS) \\
&\qquad \leq previous_{*:*:*:0:0:0}(I_{SPrice}.ITS) \\
&\sigma_{I_{INews}.ITS \in \{u | previous_{*:*:*:0:0:0}(before_{0:0:1:0:0:0}} \\
&\quad (previous_{*:*:*:0:0:0}(t))) < u \\
&\quad \leq next_{*:*:*:0:0:0}(previous_{*:*:*:0:0:0}(t))\}(I_{INews}) \\
&\bowtie_{Name=Company \wedge Price \geq Threshold} \\
&StockholderInfo
\end{aligned}$$

In the above expression, $\psi_{SPrice}(t)$ and $\psi_{INews}(t)$ are identified as follows:

$$\begin{aligned}
&\psi_{SPrice}(t) = \{u | previous_{*:*:*:0:0:0}(t) \leq u < t\}, \\
&\psi_{INews}(t) = \{u | previous_{*:*:*:0:0:0}(\\
&\quad before_{0:0:1:0:0:0}(previous_{*:*:*:0:0:0}(t))) < u \\
&\quad \leq next_{*:*:*:0:0:0}(previous_{*:*:*:0:0:0}(t))\}.
\end{aligned}$$

They satisfy the condition of Theorem 1 (ii).

In the following, we present more details of steps 4 through 6.

7.1.2 Storage Rule

A storage rule is derived for each I-sequence relation $I_i\,(1 \leq i \leq n)$ referenced in the service specification. The storage rule for I-sequence relation I_i is invoked when a delivery unit has arrived from the underlying dissemination-based information source. Then, in the condition clause, it checks whether the new data (tuple) satisfies the filtering condition shown below. If it does, it requests the mediator to store it in a temporary relation in the action clause. If I-sequence relation I_i is the master in the specification, it sets the timer alarm to raise an alarm event which invokes the relevant generation rule.

The filtering condition in the condition clause should check whether the new tuple will be used in the future. When we obtain the expression

$$\begin{aligned}
&E'(\sigma_{C_1}(\sigma_{I_1.ITS \in \psi_1(t)}(I_1)), \cdots, \\
&\quad \sigma_{C_n}(\sigma_{I_n.ITS \in \psi_n(t)}(I_n)))
\end{aligned}$$

in step 2 above,

$$\sigma_{C_i}(\sigma_{I_i.ITS \in \psi_i(t)}(I_i))$$

can be used to derive the filtering condition. For a tuple to be used in the future, it must satisfy the condition C_i. Also, it must meet the temporal condition

$$I_i.ITS \in TSet^+(\psi_i, now),$$

where "now" stands for the current time.

To summarize, the storage rule for I-sequence relation I_i can be derived as follows:

(i) Case 1: I_i is a master I-sequence relation.

Rule Storage$_i$
on: arrival(I_i)
if: $I_i.ITS \in TSet^+(\psi_i, now) \wedge C_i$
do: $Temp_{I_i}$ += I_i;
$setTimer(f(I_i.ITS), new)$;

The $setTimer(Time, Name)$ operator sets the timer alarm for the time $Time$ to raise an alarm event named $Name$. In the context of storage rules, I_i stands for the new tuple provided by the DIS wrapper rather than the I-sequence relation itself. However, we stretch this notation for simplicity.

(ii) Case 2: Otherwise.

Rule Storage$_i$
on: arrival(I_i)
if: $I_i.ITS \in TSet^+(\psi_i, now) \wedge C_i$
do: $Temp_{I_i}$ += I_i;

Storage rules derived for the service specification given in Sect. 5 are as follows. They can be reduced to those shown in Sect. 4.

Rule Storage$_{INews}$
on: arrival(I_{INews})
if: $I_{INews}.ITS \in TSet^+(\psi_{INews}, now)$
do: $Temp_{I_{INews}}$ += I_{INews};

Rule Storage$_{SPrice}$
on: arrival(I_{SPrice})
if: $I_{SPrice}.ITS \in TSet^+(\psi_{SPrice}, now)$
$\wedge I_{SPrice}.Category =' IT'$
do: $Temp_{I_{SPrice}}$ += I_{SPrice};
$setTimer(next_{*:*:*:0:0:0}(I_{SPrice}.ITS), new)$;

7.1.3 Generation Rule

A generation rule is derived from the specification. Its invocation is triggered by the alarm from the timer module set in the storage rule for the master I-sequence relation. For the expression

$$\begin{aligned}
&E'(\sigma_{C_1}(\sigma_{I_1.ITS \in \psi_1(t)}(I_1)), \\
&\quad \cdots, \sigma_{C_n}(\sigma_{I_n.ITS \in \psi_n(t)}(I_n)))
\end{aligned}$$

obtained in step 2, it requests the mediator to execute the expression

$$E'(\sigma_{Temp_{I_1}.ITS\in\psi_1(now)}(Temp_{I_1}),$$
$$\cdots,\sigma_{Temp_{I_n}.ITS\in\psi_n(now)}(Temp_{I_n})).$$

Thus, the generation rule can be specified as follows:

Rule Generation$_{new}$
on: alarm(new)
if: true
do: $O_{new} =$
$E'(\sigma_{Temp_{I_1}.ITS\in\psi_1(now)}(Temp_{I_1}),$
$\cdots,\sigma_{Temp_{I_n}.ITS\in\psi_n(now)}(Temp_{I_n}));$
$Deliver(O_{new})$.

The generation rule derived for the service specification given in Sect. 5 is as follows:

Rule Generation$_{new}$
on: alarm(new)
if: true
do: $O_{new} = \sigma_{Temp_{SPrice}.ITS\in\psi_{SPrice}(now)}$
$(Temp_{SPrice})$
$\bowtie_{CName=Name\wedge previous_{*:*:*:0:0:0}($
$before_{0:0:1:0:0:0}(I_{SPrice}.ITS))$
$\leq previous_{*:*:*:0:0:0}(I_{INews}.ITS)$
$\wedge previous_{*:*:*:0:0:0}(I_{INews}.ITS)$
$\leq previous_{*:*:*:0:0:0}(I_{SPrice}.ITS)$
$\sigma_{Temp_{INews}.ITS\in\psi_{INews}(now)}$
$(Temp_{INews})$
$\bowtie_{Name=Company\wedge Price\geq Threshold}$
$StockholderInfo;$
$Deliver(O_{new})$.

7.1.4 Garbage Disposal Rule

In the expression

$$E'(\sigma_{C_1}(\sigma_{I_1.ITS\in\psi_1(t)}(I_1)),$$
$$\cdots,\sigma_{C_n}(\sigma_{I_n.ITS\in\psi_n(t)}(I_n)))$$

obtained in step 2, if I_i may contain disposable data under ψ_i, then a garbage disposal rule for I_i is derived.

The garbage disposal rule is invoked by alarm events periodically raised from the timer module for each time interval INT. We assume that the system administrator determines the time interval INT. In the condition clause, it checks the following condition:

$$TSet^+(\psi_i, now - INT) - TSet^+(\psi_i, now) \neq \phi.$$

If it holds, it executes the action clause. In the action clause, it discards tuples which will never be used in the future, as follows:

$$Temp_{I_i} -= \sigma_{t\in TSet^+(\psi_i,\ now-INT)}$$
$$-TSet^+(\psi_i,\ now)(Temp_{I_i}).$$

Thus, if I_i may contain disposable data, a garbage disposal rule is derived as follows:

Rule GarbageDisposal$_i$
on: alarm($GarbageDisposal_i$)
if: $TSet^+(\psi_i, now - INT) - TSet^+(\psi_i, now) \neq \phi$
do: $Temp_{I_i} -= \sigma_{Temp_{I_i}.ITS\in TSet^+}$
$_{(\psi_i,\ now-INT)-TSet^+(\psi_i,\ now)}(Temp_{I_i});$
$setTimer(now + INT, GarbageDisposal_i)$.

In the example scenario, we obtain $\psi_{SPrice}(t)$ and $\psi_{INews}(t)$ for I-sequence relations I_{SPrice} and I_{INews} as shown in Sect. 7.1.1. Since both

$$TSet^+(\psi_{SPrice}, t - INT) - TSet^+(\psi_{SPrice}, t)$$
$$= \{u | previous_{*:*:*:0:0:0}(t - INT) \leq u$$
$$< previous_{*:*:*:0:0:0}(t)\},$$
$$TSet^+(\psi_{INews}, t - INT) - TSet^+(\psi_{INews}, t)$$
$$= \{u | previous_{*:*:*:0:0:0}(before_{0:0:1:0:0:0}($$
$$previous_{*:*:*:0:0:0}(t - INT))) < u$$
$$\leq previous_{*:*:*:0:0:0}(before_{0:0:1:0:0:0}($$
$$previous_{*:*:*:0:0:0}(t)))\}$$

are not always empty, they may contain disposable data. Therefore, garbage disposal rules for I_{SPrice} and I_{INews} are derived as follows:

Rule GarbageDisposal$_{SPrice}$
on: alarm($GarbageDisposal_{SPrice}$)
if: $TSet^+(\psi_{SPrice}, now - INT)$
$-TSet^+(\psi_{SPrice}, now) \neq \phi$
do: $Temp_{I_{SPrice}} -= \sigma_{Temp_{I_{SPrice}}.ITS}$
$_{\in TSet^+(\psi_{SPrice},\ now-INT)}$
$_{-TSet^+(\psi_{SPrice},\ now)}(Temp_{I_{SPrice}});$
$setTimer(now + INT,$
$GarbageDisposal_{SPrice});$

Rule GarbageDisposal$_{INews}$
on: alarm($GarbageDisposal_{INews}$)
if: $TSet^+(\psi_{INews}, now - INT)$
$-TSet^+(\psi_{INews}, now) \neq \phi$
do: $Temp_{I_{INews}} -= \sigma_{Temp_{I_{INews}}.ITS}$
$_{\in TSet^+(\psi_{INews},\ now-INT)}$
$_{-TSet^+(\psi_{INews},\ now)}(Temp_{I_{INews}});$
$setTimer(now + INT,$
$GarbageDisposal_{INews})$.

7.2 Rule Generation for Multiple Service Specifications

In the previous subsection, we assumed that we have only one service specification. In this section, we discuss the case of multiple consistent service specifications $S_1, \cdots, S_m$ defining O-sequence relations $O_1, \cdots, O_m$, respectively, on top of I-sequence relations $I_1, \cdots, I_n$. An I-sequence relation I_i may be referenced in more than one specification. In this case, an ECA rule may be used to implement multiple service specifications. Note that no conflict will occur as long as

ECA rules are generated under the proposed scheme. In the following, we assume that the expression

$$E'^j(\sigma_{C_1^j}(\sigma_{I_1.ITS\in\psi_1^j(t)}(I_1)), \cdots, \sigma_{C_n^j}(\sigma_{I_n.ITS\in\psi_n^j(t)}(I_n)))$$

is obtained by step 2 in Sect. 7.1.1 from the service specification S_j.

7.2.1 Storage Rule

A storage rule is derived for each I-sequence relation I_i that is referenced in at least one specification. The derivation is similar to that explained in Sect. 7.1.2. A difference is that the filtering condition should take care of all selection conditions

$$\sigma_{C_i^j}(\sigma_{I_i.ITS\in\psi_i^j(t)}(I_i))$$

for I_i. Another difference is that the timer alarm should be set to trigger invocation of the generation rule for each S_j that references I_i as the master I-sequence relation.

The storage rule for I_i is given as follows:

Rule Storage$_i$
on: arrival(I_i)
if: $tag(I_i, \bigvee_j(I_i.ITS \in TSet^+(\psi_i^j, now) \wedge C_i^j))$
do: $Temp_{I_i}$ += I_i;
For each S_j that references I_i as the master, if j is included in $I_i.Tag$
$setTimer(f_j(I_i.ITS), j)$.

The range of the index j in the disjunction $\bigvee_j(\ldots)$ is the set of indexes of S_j's which reference I_i. The operator $tag(I_i, \bigvee_j(\ldots))$ adds the Tag attribute to the tuple in I_i. The assigned attribute value is the set of index value j such that the tuple satisfies the condition

$$I_i.ITS \in TSet^+(\psi_i^j, now) \wedge C_i^j.$$

The operator returns the truth value of $\bigvee_j(\ldots)$.

7.2.2 Generation Rule

A generation rule is derived for each service specification S_j. In this context, each temporary relation may include tuples which are used to derive data for different services. Therefore, in the generation rule for specification S_j, we have to extract data for S_j. This can be achieved by checking the Tag attribute values.

Rule Generation$_j$
on: alarm(j)
if: true
do: $O_j = E'^j(\pi^*(\sigma_{Tag\ni j}(Temp_{I_1})), \cdots, \pi^*(\sigma_{Tag\ni j}(Temp_{I_n})))$;
$Deliver(O_j)$.

Here, π^* stands for the projection operation to eliminate the Tag attribute.

7.2.3 Garbage Disposal Rule

A garbage disposal rule is derived for I_i, if it may contain disposable data in the context of at least one service specification. Data becomes garbage when nobody is going to use it. The garbage disposal rule for I-sequence relation I_i is derived as follows:

Rule GarbageDisposal$_i$
on: alarm($GarbageDisposal_i$)
if: $(\bigcap_j(TSet^+(\psi_i^j, now - INT) - TSet^+(\psi_i^j, now))) \neq \phi$
do: $Temp_{I_i}$ −= $\sigma_{Temp_{I_i}.ITS\in\bigcap_j(TSet^+(\psi_i^j, now-INT)-TSet^+(\psi_i^j, now))}(Temp_{I_i})$;
$setTimer(now + INT, GarbageDisposal_i)$.

8. Prototype System Development

We have implemented the rule generation module on the dissemination-based information integration system *InfoWeaver* [7]. InfoWeaver was developed using Java and the rule generation module was also written as a Java program.

The system architecture of the rule generation module is shown in Fig. 3. We explain the generation procedure of ECA rules using the figure. First, steps 1 and 2 of the ECA rule generation procedure described in Sect. 7.1.1 are performed in the algebra expression transformation module. In these steps, the transformation of temporal conditional expressions involving timestamps is processed by the TS module. The selection condition for each source obtained as a result of processing in the algebra expression transformation

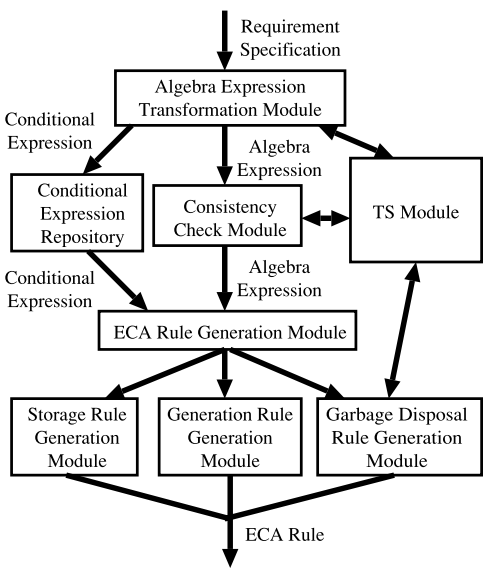

Fig. 3 Rule generation module.

module is stored in the conditional expression repository. The conditional expressions stored in the conditional expression repository are used to generate ECA rules when multiple delivery specifications are given (details are omitted). Next, the consistency check module and the TS module check the consistency of the obtained conditional expression. Finally, ECA rules are generated in each rule generation part.

We confirmed that our proposed framework actually works using the prototype rule generation module. We also verified that the rule generation module generates logically correct conditional expressions and ECA rules using the example in Sect. 2. In addition, conditional expressions and ECA rules almost equal to ψ_{SPrice} and ψ_{INews} and the ECA rules shown in Sect. 2 were generated; this implies that the transformation by the rule generation module achieves almost the same quality as manual transformation. We have made similar observations for other sample specifications.

We measured the time required to generate ECA rules using the rule generation module and some delivery requirement specifications with almost the same degree of complexity as the example given in this paper. It was shown that more than 50 percent of the rule generation time was spent in the TS module for transforming temporal conditional expressions. This means that half of the ECA rule generation time was taken in the TS module. The current rule generation module uses 14 transformation rules to transform expressions involving timestamp functions. There is room for improving the efficiency of the transformation of conditional expressions and consistency check algorithms.

9. Related Works

In this paper, we have proposed a framework for specifying new information delivery services integrating existing dissemination-based information sources and automatically deriving ECA rules from the specification.

PointCast [15] and *Castanet* [9] are examples of commercial dissemination-based information sources. Although some of those sources provide the information filtering service, they do not have the facility for information integration.

DBIS [1], [2] and *Muffin* [14] aim to extract information from multiple dissemination-based information sources. DBIS uses *Information Broker* to access multiple dissemination-based sources transparently and to select information based on user profiles. Muffin can create a new *virtual channel* based on the user profile. To select information from multiple information services, it uses frequency of delivery, freshness, and popularity as well as similarity. These two research studies only consider information selection from multiple dissemination-based information sources, and do not consider more sophisticated integration involving dissemination-based sources or other conventional information sources.

In *SADB* [18], ECA rules are used to manipulate data arriving from dissemination-based information sources. However, users have to specify ECA rules directly, and Terada et al. do not show declarative requirement specification or rule generation schemes.

OpenCQ [8] is an information integration system for distributed heterogeneous information sources. This system is based on the event-driven approach and supports continual queries. A *continual query* consists of three components, a query, a trigger condition, and a termination condition. When a trigger condition becomes true, this system repeatedly executes the query until the stop condition holds. Each time, the difference between the current query execution and the previous one is reported as a result. Liu et al.'s work is related to our approach, since it follows an event-based approach in the context of information integration. However, OpenCQ considers only a particular case of the dissemination-based delivery of the integration results. Moreover, its mediator cannot store integration results. Therefore, OpenCQ cannot support the integration of dissemination-based information sources or complex delivery requirements described in this paper. Moreover, the users have to write continual queries directly.

Tapestry [19] also supports continual execution of queries for append-only relational databases. A continual query in Tapestry is an SQL query including time conditions in the WHERE clause. Like OpenCQ, this system reports the difference from the previous result to the user. Although an append-only database can be seen as a dissemination-based information source, the queries allowed in Tapestry are limited to special cases. Moreover, Tapestry does not provide event-driven data processing or information integration facilities.

Production rules and *incremental algorithms* are proposed to be used for the maintenance of materialized views [3]–[5], [11]. Those works are related to ours, since O-sequence relations introduced in this paper could be regarded as views on top of I-sequence relations. Automatic rule derivation from view definitions is discussed in [4]. However, in the context of materialized view maintenance, Ceri and Widom do not consider temporal properties such as arrival and delivery times, temporal dependencies, or timestamp functions. Processes such as garbage disposal are not considered, either. In addition, in [4], tuples for view relations are generated as soon as base relations are updated. In contrast to this, tuples for O-sequence relations are generated at the scheduled delivery time. [11] proposes a scheme for deleting tuples in base relations which do not contribute to the materialization of view relations. The work is related to the derivation of garbage disposal rules in our scheme. In our approach, each tuple in I-sequence relations is available only after its arrival time. In addition, tuples to be materialized in O-sequence relations

change dynamically. However, Garcia-Molina et al. do not consider such temporal properties.

SEQ [17] is a data model for sequence data. Sequence relations in this paper also model sequences of delivery units. SEQ is based on the relational model, but a number of new operators are introduced. In this paper, we have slightly extended the original relational algebra with operators in order to deal with timestamps.

10. Conclusion

In this paper, we have discussed a scheme for declaratively specifying new information delivery services integrating multiple dissemination-based information sources. We have assumed the mediator/wrapper-based information integration system architecture, in which ECA rules are employed to implement event-driven data storage, integration, and delivery operations. We have also shown how to derive ECA rules from the service specifications.

The proposed scheme is being implemented on the prototype InfoWeaver system. Future research issues are as follows:

1. The algebraic specification provides a sound basis for more declarative and user-friendly specification schemes. They include schemes based on SQL and GUI. Development of such facilities is one of the important future research issues.
2. We need to study the improvement of specification schemes and ECA derivation schemes to cope with more complex service requirements. Specification of user preferences on information is an interesting research topic.
3. In the proposed approach, we have introduced five basic timestamp functions. Their expressive power requires further analysis.
4. The concept of consistency introduced in the paper takes only temporal integrity into consideration. We can expand the approach to incorporate data arrival properties of the underlying dissemination-based information sources.
5. We have to deal with cases where the scheduled delivery time must be determined in a more sophisticated manner.

Acknowledgments

The authors are grateful to Mr. Yousuke Watanabe for his contribution to the prototype system development. This research is supported in part by a Grant-in-Aid for Scientific Research from JSPS and MEXT, Japan.

References

[1] D. Aksoy, M. Altinel, R. Bose, U. Cetintemel, M. Franklin, J. Wang, and S. Zdonik, "Research in data broadcast and dissemination," Proc. AMCP '98, pp.194–207, 1998.
[2] M. Altinel, D. Aksoy, T. Baby, M. Franklin, W. Shapiro, and S. Zdonik, "DBIS toolkit — Adaptable middleware for large-scale data delivery," Proc. ACM SIGMOD '99, pp.544–546, 1999.
[3] J.A. Blakeley, P.A. Larson, and F.W. Tompa, "Efficiently updating materialized views," Proc. ACM SIGMOD '86, pp.61–71, May 1986.
[4] S. Ceri and J. Widom, "Deriving production rules for incremental view maintenance," Proc. 17th VLDB, pp.577–589, 1991.
[5] E.N. Hanson, "A performance analysis of view materialization strategies," Proc. ACM SIGMOD '87, pp.440–453, 1987.
[6] H. Kitagawa, T. Kajino, and Y. Ishikawa, "Algebraic service specification and rule generation for integrating multiple dissemination-based information sources," Proc. 7th International Conference on Database Systems for Advanced Applications, pp.344–351, 2001.
[7] H. Kitagawa, A. Morishima, and H. Mizuguchi, "Integration of heterogeneous information sources in InfoWeaver," in Advances in Database and Multimedia for the New Century — A Swiss/Japanese Perspective, pp.124–137, World Scientific Publishing, 2000.
[8] L. Liu, C. Pu, and W. Tang, "Continual queries for Internet scale event-driven information delivery," IEEE Trans. Knowl. Data Eng., vol.11, no.4, pp.610–628, 1999.
[9] Marimba Inc., Castanet, http://www.marimba.com/products/castanet-intro.htm
[10] H. Mizuguchi, H. Kitagawa, Y. Ishikawa, and A. Morishima, "A rule-oriented architecture to incorporate dissemination-based information delivery into information integration environments," Proc. 2000 ADBIS-DASFAA Symposium on Advances in Databases and Information Systems, pp.185–199, 2000.
[11] H. Garcia-Molina, W.J. Labio, and J. Yang, "Expiring data in a warehouse," Proc. 24th VLDB, pp.500–511, 1998.
[12] A. Morishima and H. Kitagawa, "InfoWeaver: Dynamic and Tailor-made integration of structured documents, Web, and databases," Proc. ACM DL '99, pp.235–236, 1999.
[13] N.W. Paton and O. Diaz, "Active database systems," ACM Comp. Serv., vol.31, no.1, pp.63–103, 1999.
[14] M. Qiang, H. Kondo, K. Sumiya, and K. Tanaka, "Virtual TV channel: Filtering merging and presenting Internet broadcasting channels," ACM DL Workshop on WOWS, 1999.
[15] S. Ramakrishnan and V. Dayal, "The PointCast network," ACM SIGMOD Record, vol.27, no.2, p.520, 1998.
[16] P.G. Selinger, M.M. Astrahan, D.D. Chamberlin, R.A. Lorie, and T.G. Price, "Access path selection in a relational database management system," Proc. ACM SIGMOD '79, pp.23–34, 1979.
[17] P. Seshadri, M. Livny, and R. Ramakrishnan, "SEQ: A model for sequence databases," Proc. ICDE, pp.232–239, 1995.
[18] T. Terada, M. Tsukamoto, and S. Nishio, "Design and implementation of an active database system for receiving broadcast data," IEICE Trans. Inf. & Syst. (Japanese Edition), vol.J83-D-I, no.12, pp.1272–1283, Dec. 2000.
[19] D.B. Terry, D. Goldberg, D. Nichols, and B.M. Oki, "Continuous queries over append-only databases," Proc. SIGMOD '92, pp.321–330, 1992.

Tomoyuki Kajino received the B.E. and M.E. degrees in information engineering from the University of Tsukuba in 1999 and 2001, respectively. He joined Beacon Information Technology Inc. in 2001, where he has been engaged in research and development related to database and Web technologies. He is a member of IPSJ.

Hiroyuki Kitagawa received the B.Sc. degree in physics and the M.Sc. and Dr.Sc. degrees in computer science, all from the University of Tokyo, in 1978, 1980, and 1987, respectively. He joined the Institute of Information Sciences and Electronics, University of Tsukuba, in 1988, where he is currently a professor. He was a visiting scholar of the University of Maryland from 1984 to 1985, and of Carnegie Mellon University from 2001 to 2002. His research interests include information integration, WWW and databases, XML databases, and data mining. He is chair of ACM SIGMOD Japan Chapter, and a member of ACM, the IEEE Computer Society, DBSJ, IPSJ, and JSSST.

Yoshiharu Ishikawa received the B.E., M.E. and Dr.Eng. degrees in information engineering from the University of Tsukuba in 1989, 1991, and 1995, respectively. From 1994 to 1999, he was a research associate at Nara Institute of Science and Technology. He joined the Institute of Information Sciences and Electronics, University of Tsukuba, in 1999, where he is currently an associate professor. His research interests include document databases and XML technologies, spatio-temporal databases, information retrieval, and Web technologies. He is a member of ACM, the IEEE Computer Society, DBSJ, IPSJ, and ACM SIGMOD Japan Chapter.